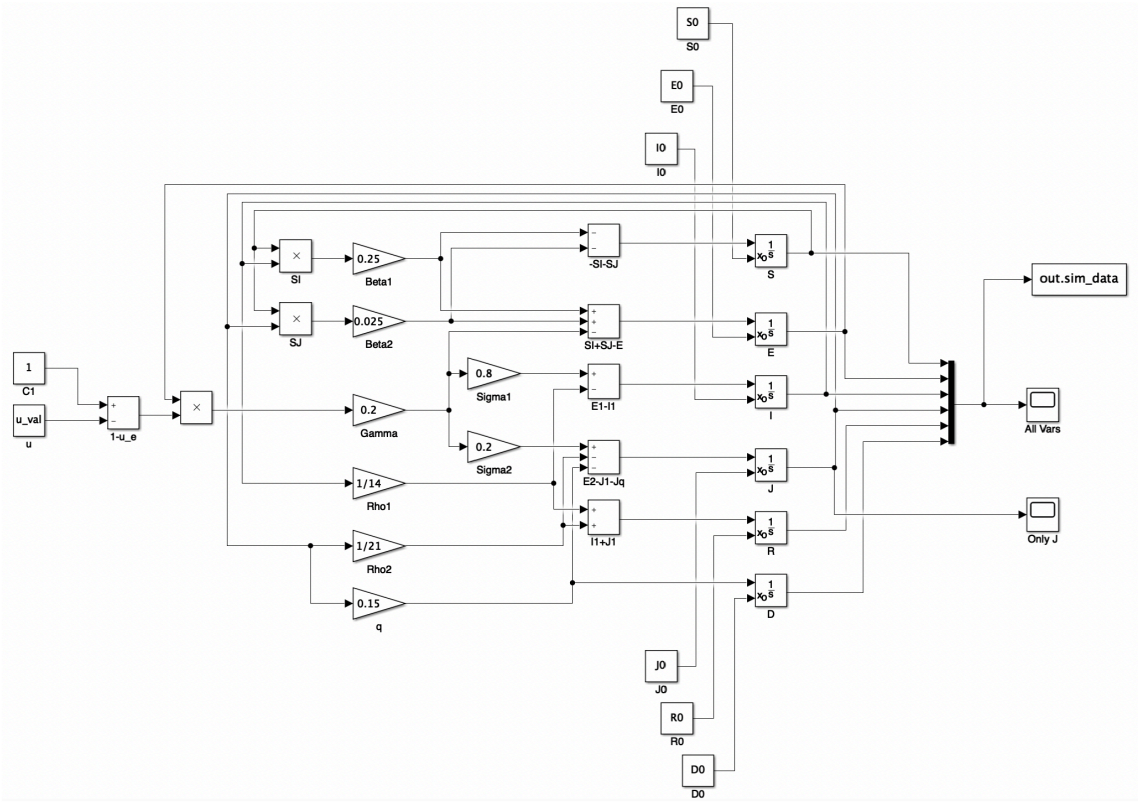
ECE141 Project (S20)

Problem 1

Let the S, E, I, J, R, and D range between 0 and 1 so that their values can be interpreted as the fraction of the population in the respective class. Simulate the model in simulink for different initial conditions and comment on the observed behavior.

```
imshow(imread("Non_linear_model.bmp"))
```

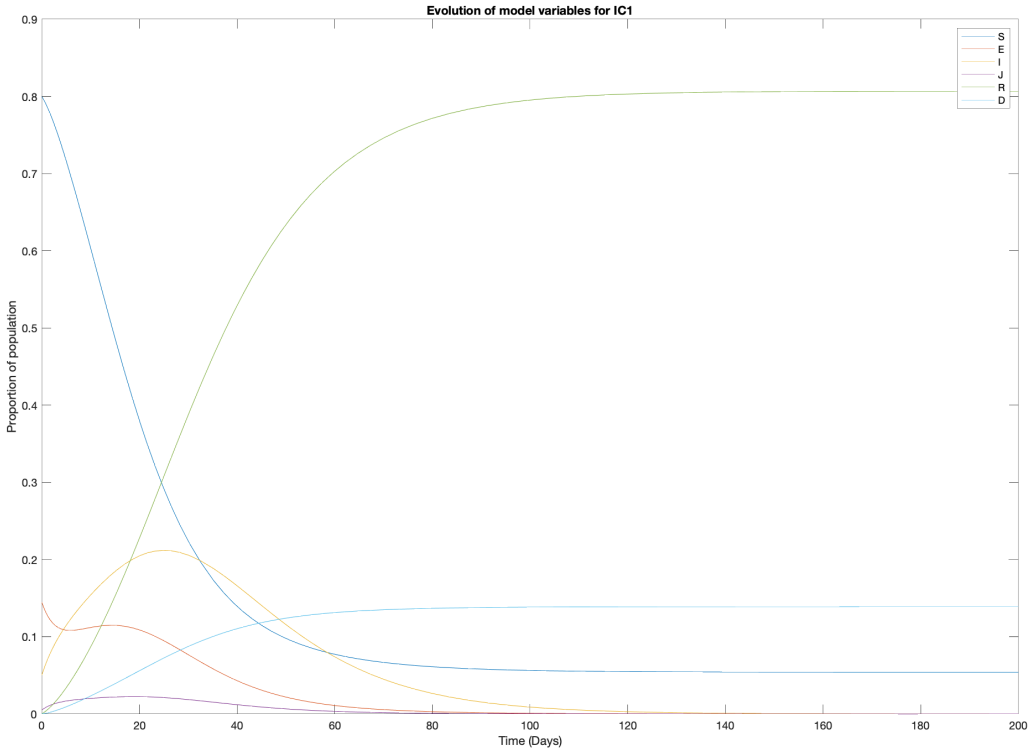


Initial conditions 1:

$S(0) = 0.8,\ E(0) = 0.145,\ I(0) = 0.05,\ J(0) = 0.005,\ R(0) = 0,\ D(0) = 0$

```
%Set up initial constants
sim_t = 200.0;
S0 = 0.8;
I0 = 0.05;
E0 = 0.145;
J0 = 0.005;
```

```
R0 = 0;
D0 = 0;
u_val = 0;

% Simulate model with given ICs
simOut = sim('Non_linear_model',sim_t);
sim_data1 = simOut.get("sim_data");

% Get data and plot
data = sim_data1.Data;
t1 = sim_data1.Time;

plot(t1, data)
title("Evolution of model variables for IC1")
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("S","E","I","J","R","D")
```




```
fprintf("Steady state S = %f",data(end,1))
```

```
Steady state S = 0.054177
```

```
fprintf("Steady state R = %f",data(end,5))
```

```
Steady state R = 0.806759
```

```
fprintf("Steady state D = %f",data(end,6))
```

```
Steady state D = 0.139027
```

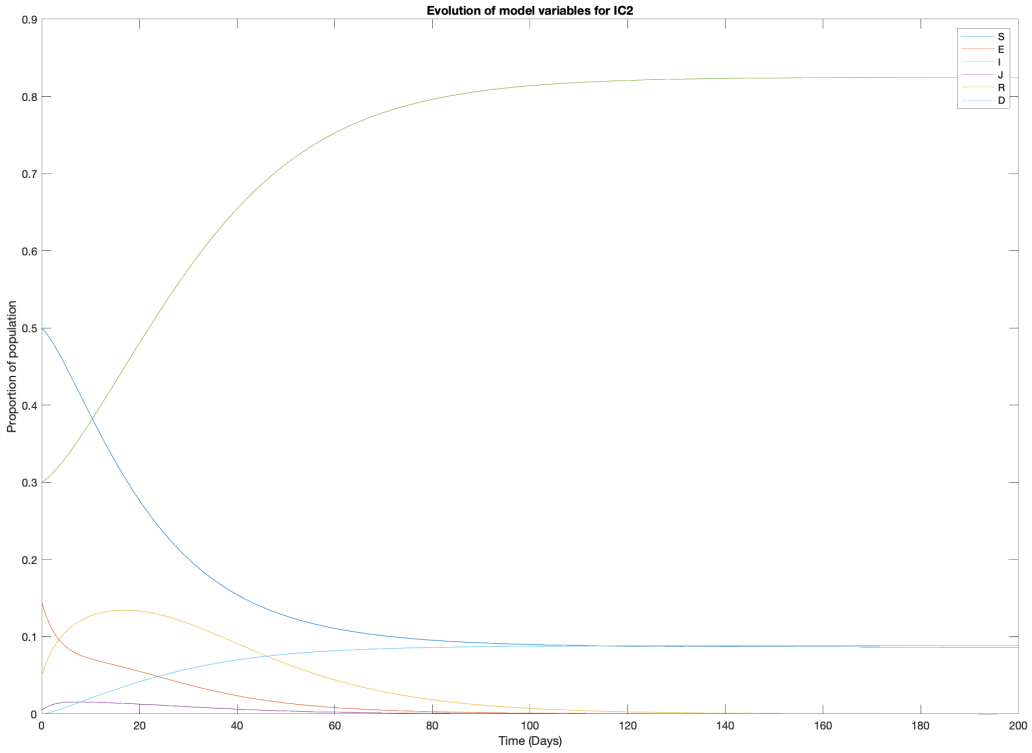
Given these particular initial conditions the steady state of the system shows that approximately 80.6% of the population gets COVID-19 and recovers from the disease, while 5.4% never contracts it and 13.9% dies.

Initial conditions 2

Another interesting initial condition would be to increase the number of recovered people, in order to test the effect of herd immunity:

$S(0) = 0.5,\ E(0) = 0.145,\ I(0) = 0.05,\ J(0) = 0.005,\ R(0) = 0.3,\ D(0) = 0$

```
% Modify initial conditions
S0 = 0.5;
R0 = 0.3;

% Simulate model with given ICs
simOut = sim('Non_linear_model',sim_t);
sim_data2 = simOut.get("sim_data");

% Get data and plot
data = sim_data2.Data;
t2 = sim_data2.Time;

plot(t2, data)
title("Evolution of model variables for IC2")
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("S","E","I","J","R","D")
```



```
fprintf("Steady state S = %f",data(end,1))
```

```
Steady state S = 0.086629
```

```
fprintf("Steady state R = %f",data(end,5))
```

```
Steady state R = 0.824746
```

```
fprintf("Steady state D = %f",data(end,6))
```

```
Steady state D = 0.088557
```

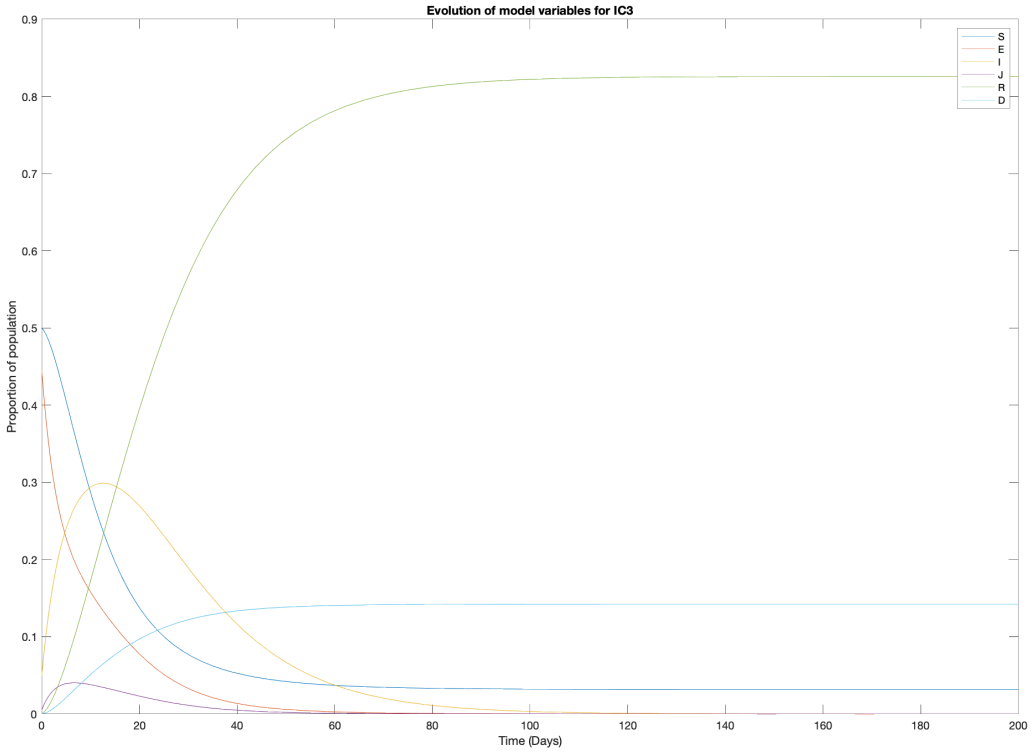
These new initial conditions justify the importance of herd immunity when fighting against a widespread pandemic, as the steady state proportion of recovered people has increased by less than 0.02 (2.5% increase from original results), while deaths have dropped by 0.05 (36% decrease from original results)

Initial conditions 3

Another relevant initial condition would be a situation with a greater proportion of the population being initially exposed to the virus, as in the early stages of the pandemic most countries took almost no measures to prevent exposure.

$S(0) = 0.5,\ E(0) = 0.445,\ I(0) = 0.05,\ J(0) = 0.005,\ R(0) = 0,\ D(0) = 0$

```
% Modify initial conditions
R0 = 0;
E0 = 0.445;

% Simulate model with given ICs
simOut = sim('Non_linear_model',sim_t);
sim_data3 = simOut.get("sim_data");

% Get data and plot
data = sim_data3.Data;
t3 = sim_data3.Time;

plot(t3, data)
title("Evolution of model variables for IC3")
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("S","E","I","J","R","D")
```




```
fprintf("Steady state S = %f",data(end,1))
```

```
Steady state S = 0.031781
```

```
fprintf("Steady state R = %f",data(end,5))
```

```
Steady state R = 0.825783
```

```
fprintf("Steady state D = %f",data(end,6))
```

```
Steady state D = 0.142428
```

It is interesting to note that the proportion of recovered population has increased in a smiliar manner to the previous scenario, and the proportion of deaths is slightly higher than that of the original configuration. The reason for these increases is that more people contract the disease at some point in time, given that at steady state only a proportion of 0.03 is still suceptible. However, as explored in problem 2 the most important difference between initial conditions 2 and 3 is the shape of the J curve, which indicates the proportion of population that is seriously ill at any given time.

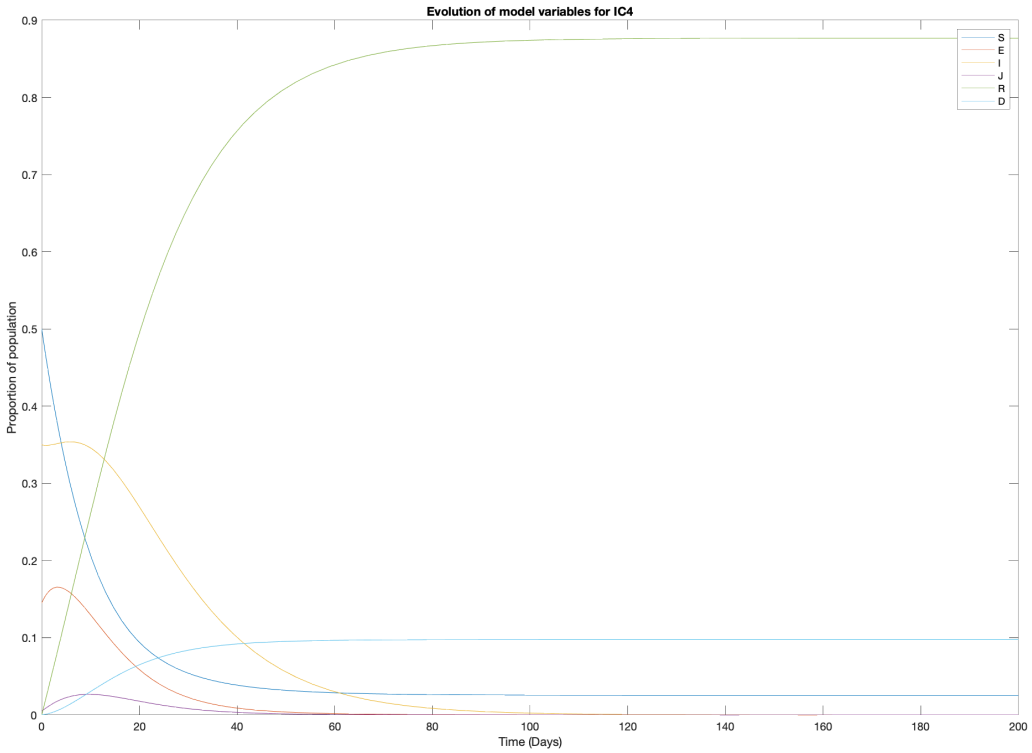
Initial condition 4

Another important initial condition would be a situation with a greater initial proportion of asymptomatic carriers (also known as Infected in the mathematical model)

$S(0) = 0.5,\ E(0) = 0.145,\ I(0) = 0.35,\ J(0) = 0.005,\ R(0) = 0,\ D(0) = 0$

```
% Modify initial conditions
E0 = 0.145;
```

```
I0 = 0.35;

% Simulate model with given ICs
simOut = sim('Non_linear_model',sim_t);
sim_data4 = simOut.get("sim_data");

% Get data and plot
data = sim_data4.Data;
t4 = sim_data4.Time;

plot(t4, data)
title("Evolution of model variables for IC4")
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("S","E","I","J","R","D")
```




```
fprintf("Steady state S = %f",data(end,1))
```

```
Steady state S = 0.025501
```

```
fprintf("Steady state R = %f",data(end,5))
```

```
Steady state R = 0.876655
```

```
fprintf("Steady state D = %f",data(end,6))
```

```
Steady state D = 0.097840
```

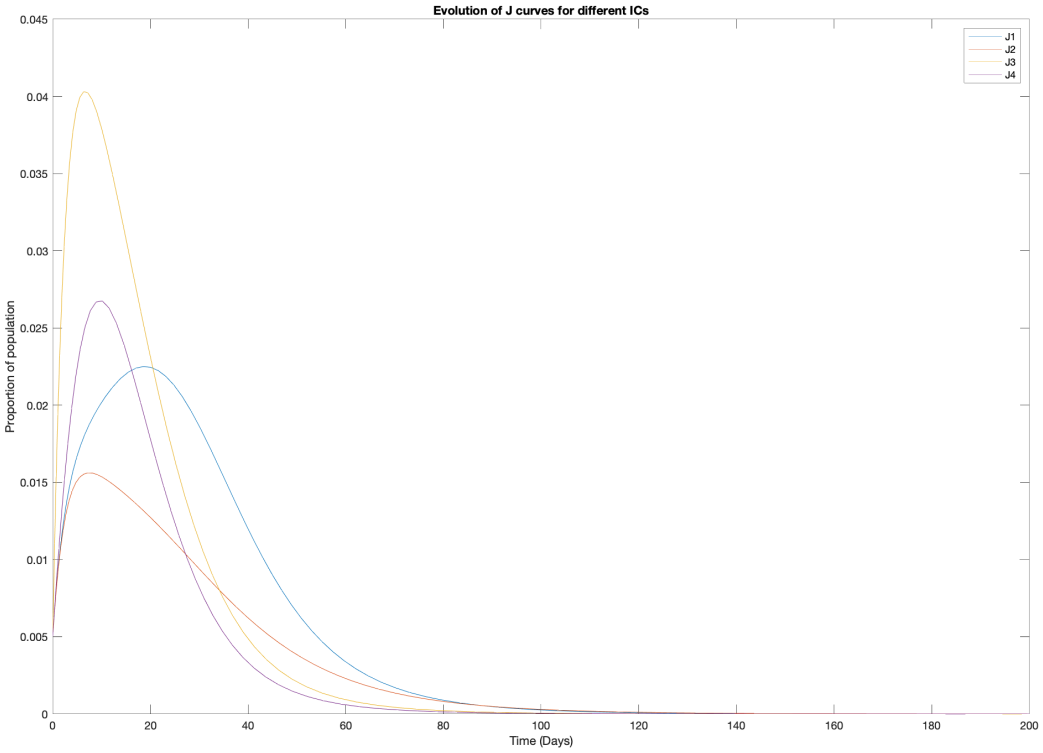
Counterintuitevely, the sharp increase in initially infected people leads to a decrease in steady state value of D, even as the final value of S is the lowest one of all possible configurations (as most people contracted the virus). A possible reason for this is that the mathematical model does not consider the case of initially asymptomatic people who later develop a serious illness, given that population can only move to de D state by going through J. These types of results indicate some of the limitations of the chosen mathematical model.

Problem 2

Comment on the number of hospital beds required to treat all the population in class J.

For the initial conditions analyzed in the previous problem we mainly focused on the steady state value of D, R, and S, as these describe the final state of the population within the mathematical model. However, in real world situations, the shape of the J curve is significantly more important, as healthcare systems are only equipped to handle a certain amount of seriously ill people simultaneously. Therefore, this section will focus on how those initial conditions affected the J curve, and what conclusions can be drawn from the results.

```
% Obtain J data
J1 = sim_data1.Data(:,4);
J2 = sim_data2.Data(:,4);
J3 = sim_data3.Data(:,4);
J4 = sim_data4.Data(:,4);

% Plot
plot(t1, J1)
hold on
plot(t2, J2)
plot(t3, J3)
plot(t4, J4)
hold off
title("Evolution of J curves for different ICs")
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("J1", "J2", "J3", "J4")
```




```
max(J1*100)
```

```
ans = 2.2502
```

```
max(J2*100)
```

```
ans = 1.5617
```

```
max(J3*100)
```

```
ans = 4.0304
```

```
max(J4*100)
```

```
ans = 2.6747
```

For all the analyzed initial conditions we can clearly see that the maximum amount of ICUs requried to treat seriously ill patients can vary significantly. This factor is vital when analyzing the consequences of COVID-19 as countries only have a limited amount of ICUs available, and very sharp peaks can lead to massive increases in mortality rates from patients not being able to obtain proper treatment.

The observed peaks were at 2.5%, 1.56%, 4% and 2.67%. It is interesting to note that not even 30% of herd immunity (case 2) could reduce the peak below 1.5%, and even countries with highly advanced medical infrastructure do not have the resources to treat these many patients simultaneously. For the purposes of later problems we will assume the country being analyzed has enoguh ICUs to treat 1% of its population simultaneously, even though this number will be substantially lower in many real-world scenarios.

Using our assumption we can see that even in the best case analyzed more than a third of seriously ill patients will not be able to obtain proper medical treatment, and this means that mortality rate will not only increase for these individuals, but also for any other person that experiences medical complications requiring intensive treatment and not related to COVID-19.

In conclusion, given that it might be difficult to greately affect the deaths caused by the virus in the long run, without the introduction of new treatments like a vaccine, the mathematical model clearly indicates that our social behavior is instrumental in reducing the death toll of COVID-19 by modifying the shape of the J curve.

Problem 3

Compute the equilibrium pairs for this system of nonlinear differential equations. Show the linearization of this model at an equilibrium pair cannot be used for the purposes of control. This illustrates the limits of the techniques you learned in this class and in this project we will use a more advanced technique (switched linear systems) for control purposes.

Compute the equilibrium pairs for non-linear system with added input $0 \leq u < 1$:

(X_e, u_e) is an equilibrium pair $\iff \dot{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$\dot{D} = 0 = \text{qJ} \iff J = 0$

$\dot{R} = 0 = \rho_1 I + \rho_2 J = \rho_1 I \iff I = 0$

$\dot{J} = 0 = (1-u)\sigma_2 \gamma E - \rho_2 J - \text{qJ} = (1-u)\sigma_2 \gamma E \iff E = 0 \text{ or } u = 1, \text{but problem states that } u = 1 \text{ is impossible} \Rightarrow E = 0$

$\dot{I} = 0 = (1-u)\sigma_1 \gamma E - \rho_1 I = (1-u)\sigma_1 \gamma E \iff \text{Already satisfied with previous constraints}$

$\dot{E} = 0 = \beta_1 \text{SI} + \beta_2 \text{SJ} - (1-u)\gamma\, E \iff \text{Already satisfied with previous constraints}$

$\dot{S} = 0 = -\beta_1 \text{SI} - \beta_2 \text{SJ} \iff \text{Already satisfied with previous constraints}$

Total population constraint: $1 = S + E + I + J + R + D = S + R + D$

Full set of constraints for equilibrium pair:

$I = J = E = 0, \quad S + R + D = 1, \quad 0 \leq u < 1$

Define given set of differential equations as matrices

$$f(X,u) = \dot{X} = \begin{bmatrix} \dot{S} \\ \dot{E} \\ \dot{I} \\ \dot{J} \\ \dot{R} \\ \dot{D} \end{bmatrix} = \begin{bmatrix} -\beta_1 \mathrm{SI} - \beta_2 \mathrm{SJ} \\ \beta_1 \mathrm{SI} + \beta_2 \mathrm{SJ} - (1-u)\gamma E \\ (1-u)\sigma_1 \gamma E - \rho_1 I \\ (1-u)\sigma_2 \gamma E - \rho_2 J - \mathrm{qJ} \\ \rho_1 I + \rho_2 J \\ \mathrm{qJ} \end{bmatrix}$$

```
% Define symbolic variables
syms S E I J R D u;
syms beta1 beta2 gamma sigma1 sigma2 rho1 rho2 q

% Define Non-linear derivative matrix
Non_lin = [-beta1*S*I-beta2*S*J;
            beta1*S*I+beta2*S*J-(1-u)*gamma*E;
            (1-u)*sigma1*gamma*E-rho1*I;
            (1-u)*sigma2*gamma*E-rho2*J-q*J;
            rho1*I+rho2*J;
            q*J];
disp(Non_lin)
```

$$\begin{pmatrix} -\mathrm{I}\,S\,\beta_1 - J\,S\,\beta_2 \\ \mathrm{I}\,S\,\beta_1 + J\,S\,\beta_2 + \mathrm{E}\,\gamma\,(u-1) \\ -\mathrm{I}\,\rho_1 - \mathrm{E}\,\gamma\,\sigma_1\,(u-1) \\ -J\,q - J\,\rho_2 - \mathrm{E}\,\gamma\,\sigma_2\,(u-1) \\ \mathrm{I}\,\rho_1 + J\,\rho_2 \\ J\,q \end{pmatrix}$$

Calculate derivative of non-linear relationship matrix with respect to state vector

$$\frac{\partial f(X,u)}{\partial X} = \begin{bmatrix} \frac{\partial \dot{S}}{\partial S} & \frac{\partial \dot{S}}{\partial E} & \frac{\partial \dot{S}}{\partial I} & \frac{\partial \dot{S}}{\partial J} & \frac{\partial \dot{S}}{\partial R} & \frac{\partial \dot{S}}{\partial D} \\ \frac{\partial \dot{E}}{\partial S} & \frac{\partial \dot{E}}{\partial E} & \frac{\partial \dot{E}}{\partial I} & \frac{\partial \dot{E}}{\partial J} & \frac{\partial \dot{E}}{\partial R} & \frac{\partial \dot{E}}{\partial D} \\ \frac{\partial \dot{I}}{\partial S} & \frac{\partial \dot{I}}{\partial E} & \frac{\partial \dot{I}}{\partial I} & \frac{\partial \dot{I}}{\partial J} & \frac{\partial \dot{I}}{\partial R} & \frac{\partial \dot{I}}{\partial D} \\ \frac{\partial \dot{J}}{\partial S} & \frac{\partial \dot{J}}{\partial E} & \frac{\partial \dot{J}}{\partial I} & \frac{\partial \dot{J}}{\partial J} & \frac{\partial \dot{J}}{\partial R} & \frac{\partial \dot{J}}{\partial D} \\ \frac{\partial \dot{R}}{\partial S} & \frac{\partial \dot{R}}{\partial E} & \frac{\partial \dot{R}}{\partial I} & \frac{\partial \dot{R}}{\partial J} & \frac{\partial \dot{R}}{\partial R} & \frac{\partial \dot{R}}{\partial D} \\ \frac{\partial \dot{D}}{\partial S} & \frac{\partial \dot{D}}{\partial E} & \frac{\partial \dot{D}}{\partial I} & \frac{\partial \dot{D}}{\partial J} & \frac{\partial \dot{D}}{\partial R} & \frac{\partial \dot{D}}{\partial D} \end{bmatrix}$$

```
lin_x = sym(zeros([6,6]));
for i=1:6
    lin_x(i,1) = diff(Non_lin(i),S);
    lin_x(i,2) = diff(Non_lin(i),E);
```

```
    lin_x(i,3) = diff(Non_lin(i),I);
    lin_x(i,4) = diff(Non_lin(i),J);
    lin_x(i,5) = diff(Non_lin(i),R);
    lin_x(i,6) = diff(Non_lin(i),D);
end
disp(lin_x)
```

$$\begin{pmatrix} -\mathrm{I}\,\beta_1 - J\,\beta_2 & 0 & -S\,\beta_1 & -S\,\beta_2 & 0 & 0 \\ \mathrm{I}\,\beta_1 + J\,\beta_2 & \gamma\,(u-1) & S\,\beta_1 & S\,\beta_2 & 0 & 0 \\ 0 & -\gamma\,\sigma_1\,(u-1) & -\rho_1 & 0 & 0 & 0 \\ 0 & -\gamma\,\sigma_2\,(u-1) & 0 & -q-\rho_2 & 0 & 0 \\ 0 & 0 & \rho_1 & \rho_2 & 0 & 0 \\ 0 & 0 & 0 & q & 0 & 0 \end{pmatrix}$$

Calculate derivative of non-linear relationship with respect to input vector

$$\frac{\partial f(X,u)}{\partial u} = \begin{bmatrix} \frac{\partial \dot{S}}{\partial u} \\ \frac{\partial \dot{E}}{\partial u} \\ \frac{\partial \dot{I}}{\partial u} \\ \frac{\partial \dot{J}}{\partial u} \\ \frac{\partial \dot{R}}{\partial u} \\ \frac{\partial \dot{D}}{\partial u} \end{bmatrix}$$

```
lin_u = sym(zeros([6,1]));
for i=1:6
    lin_u(i,1) = diff(Non_lin(i),u);
end
disp(lin_u)
```

$$\begin{pmatrix} 0 \\ \mathrm{E}\,\gamma \\ -\mathrm{E}\,\gamma\,\sigma_1 \\ -\mathrm{E}\,\gamma\,\sigma_2 \\ 0 \\ 0 \end{pmatrix}$$

Choose specific equilibrium point and evaluate calculated derivatives

$I = J = E = 0,\ \ S = 0.8,\ \ R = 0.1,\ \ D = 0.1\,,\ \ u = 0.5$

```
% Parameter definition
```

```
beta1 = 0.25;
beta2 = 0.025;
gamma = 0.2;
sigma1 = 0.8;
sigma2 = 0.2;
rho1 = 1/14;
rho2 = 1/21;
q = 0.15;
% Equilibrium point value definition
E = 0;
I = 0;
J = 0;
s_eval_lin_x = subs(lin_x);
R = 0.1;
D = 0.1;
S = 0.8;
u = 0.5;

% Substitute
verif_equ = subs(Non_lin);
eval_lin_x = subs(lin_x);
eval_lin_u = subs(lin_u);
% Verify that chosen values generate an equilibrium point
disp(verif_equ)
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

```
% Display evaluated derivative matrices
disp(eval_lin_x)
```

$$\begin{pmatrix} 0 & 0 & -\frac{1}{5} & -\frac{1}{50} & 0 & 0 \\ 0 & -\frac{1}{10} & \frac{1}{5} & \frac{1}{50} & 0 & 0 \\ 0 & \frac{2}{25} & -\frac{1}{14} & 0 & 0 & 0 \\ 0 & \frac{1}{50} & 0 & -\frac{83}{420} & 0 & 0 \\ 0 & 0 & \frac{1}{14} & \frac{1}{21} & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{20} & 0 & 0 \end{pmatrix}$$

```
disp(eval_lin_u)
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

This final results shows why the results obtained from the linearazied model are useless for control purposes, independently of the equilibrium pair chosen, as one of the constraints establishes $E = 0$. This zeroes out all terms in the input derivative matrix, indicating that the controler would have no effect on the system.

$$\frac{\partial f(X,u)}{\partial X} \text{ evaluated at } (X_e, u_e) = A$$

$$\frac{\partial f(X,u)}{\partial u} \text{ evaluated at } (X_e, u_e) = B$$

$$\frac{\partial \delta X}{\partial t} = A\delta X + B\delta u = A\delta X \iff \text{Controller has no effect on system output}$$

Problem 4

Our objective is keep the number of infected people requiring hospitalization (class J) small. Using the linearized model, compute the transfer function from J(0) to J. How do the poles of this transfer function depend on u?

```
% Define symbolic functions and linearized relationship matrix
syms E_delta(t) I_delta(t) J_delta(t) S_e u_e s;
S = S_e;
u = u_e;
disp(subs(s_eval_lin_x))
```

$$\begin{pmatrix} 0 & 0 & -\frac{S_e}{4} & -\frac{S_e}{40} & 0 & 0 \\ 0 & \frac{u_e}{5} - \frac{1}{5} & \frac{S_e}{4} & \frac{S_e}{40} & 0 & 0 \\ 0 & \frac{4}{25} - \frac{4\,u_e}{25} & -\frac{1}{14} & 0 & 0 & 0 \\ 0 & \frac{1}{25} - \frac{u_e}{25} & 0 & -\frac{83}{420} & 0 & 0 \\ 0 & 0 & \frac{1}{14} & \frac{1}{21} & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{20} & 0 & 0 \end{pmatrix}$$

```
% Define variable assumptions
assume([u_e, S_e, t] > 0)
assume(u_e < 1)
```

```
% Define derivatives of symbolic functions
dI = diff(I_delta,t);
dE = diff(E_delta,t);
dJ = diff(J_delta,t);

% Define symbolic differential equations
eqn1 = dI == 4/25*(1-u_e)*E_delta-1/14*I_delta;
disp(eqn1)
```

$$\frac{\partial}{\partial t} I_\delta(t) = -\frac{I_\delta(t)}{14} - E_\delta(t) \left(\frac{4\,u_e}{25} - \frac{4}{25}\right)$$

```
eqn2 = dE == -1/5*(1-u_e)*E_delta+S_e/4*I_delta+S_e/40*J_delta;
disp(eqn2)
```

$$\frac{\partial}{\partial t} E_\delta(t) = E_\delta(t) \left(\frac{u_e}{5} - \frac{1}{5}\right) + \frac{S_e\, I_\delta(t)}{4} + \frac{S_e\, J_\delta(t)}{40}$$

```
eqn3 = dJ == 1/25*(1-u_e)*E_delta-83/420*J_delta;
disp(eqn3)
```

$$\frac{\partial}{\partial t} J_\delta(t) = -\frac{83\, J_\delta(t)}{420} - E_\delta(t) \left(\frac{u_e}{25} - \frac{1}{25}\right)$$

```
% Take Laplace transform
eqn1LT = laplace(eqn1,t,s);
eqn2LT = laplace(eqn2,t,s);
eqn3LT = laplace(eqn3,t,s);

% Substitute symbolic Laplace transforms for new variables
syms I_LT J_LT E_LT;
eqn1LT = subs(eqn1LT, [laplace(I_delta,t,s), ...
    laplace(E_delta,t,s),laplace(J_delta,t,s)],[I_LT,E_LT,J_LT]);
eqn2LT = subs(eqn2LT, [laplace(I_delta,t,s), ...
    laplace(E_delta,t,s),laplace(J_delta,t,s)],[I_LT,E_LT,J_LT]);
eqn3LT = subs(eqn3LT, [laplace(I_delta,t,s), ...
    laplace(E_delta,t,s),laplace(J_delta,t,s)],[I_LT,E_LT,J_LT]);

% Solve system
eqns = [eqn1LT eqn2LT eqn3LT];
vars = [I_LT J_LT E_LT];
[I_LT, J_LT, E_LT] = solve(eqns,vars);
% Substitute 0 for undesired initial conditions
vars = [E_delta(0) I_delta(0)];
values = [0 0];
J_LT = subs(J_LT,vars,values);

% Simplify and display
H = simplify(J_LT/J_delta(0));
disp(H)
```

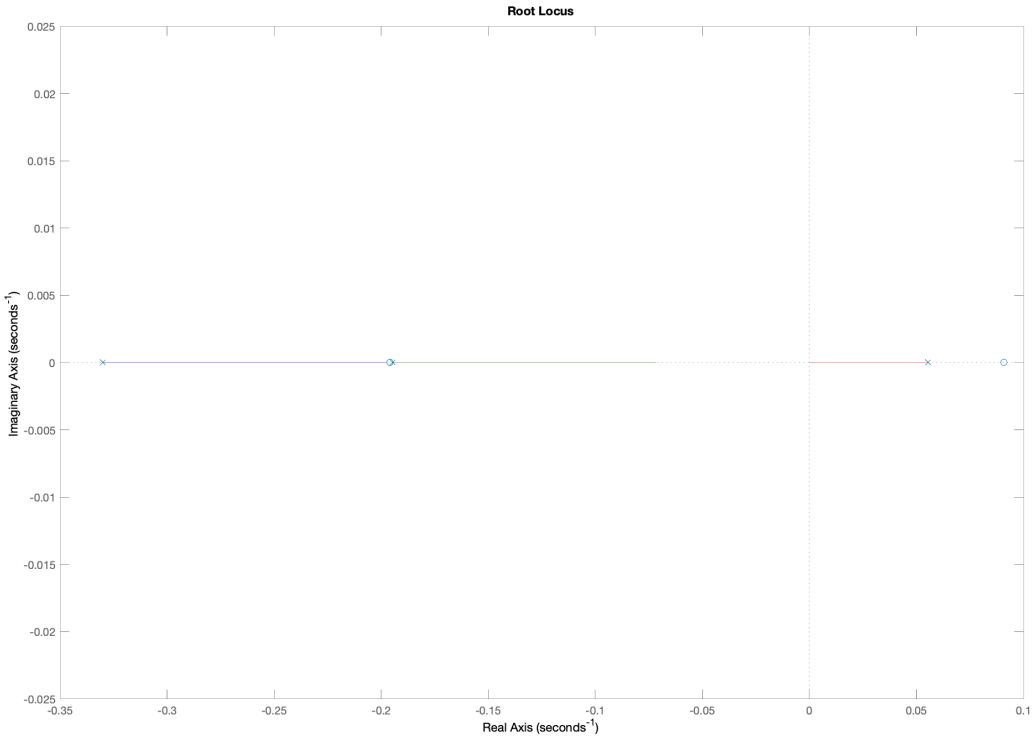
$$\frac{79800\,s - 11760\,S_e - 4200\,u_e + 11760\,S_e\,u_e - 58800\,s\,u_e + 294000\,s^2 + 4200}{19970\,s - 2345\,S_e - 830\,u_e - 12054\,S_e\,s + 2345\,S_e\,u_e - 15820\,s\,u_e - 58800\,s^2\,u_e + 137900\,s^2 + 294000\,s^3 -}$$

After finding the transfer function of the form:

$$H(s) = \frac{b(s)}{a(s)} = \frac{b(s)}{1 + u_e\,L(s)}$$

$$H(s) = \frac{294000s^2 + (79800 - 58800\,u_e)s + (4200 - 4200\,u_e + 11760\,S_e\,u_e - 11760\,S_e)}{1 + u_e\,\dfrac{-58800s^2 + (S_e\,12054 - 15820)s + (S_e\,2345 - 830)}{294000s^3 + 137900s^2 + (19970 - 12054\,S_e)s + (830 - 2345\,S_e)}}$$

```
% Define arbitrary equilibrium S to obtain Root Locus
S_e = 0.8;
num = [-58800 (-15820+S_e*12054) (-830+2345*S_e)];
denom = [294000 137900 (19970-12054*S_e) (830-2345*S_e)];

% Define 0 < k < 1
k = linspace(0,1,1000);

% Define transfer function and plot
L = tf(num,denom);
rlocus(L,k)
```



The Root locus plot seems suggests that increasing the value of u_e slows the overall growth of the system, but the final result cannot be made stable with the value chosen for S_e. This result might not make intuitive sense

at first, as the non-linear model was clearly established to be stable. However, the result can be justified by analyzing the limitations of the linearized model.

If we look back the the non-linear model, the evolution of the J curve is solely determined by the ratio between exposed individuals becoming seriously ill and seriously ill patients recovering or dying. In Problem 1 we fixed the recovery and death rate for individuals in class J, so this provides a well defined boundary for stability: the system will be stable if and only if the amount of exposed individuals becoming seriously ill drops below this combined removal rate at some point in time.

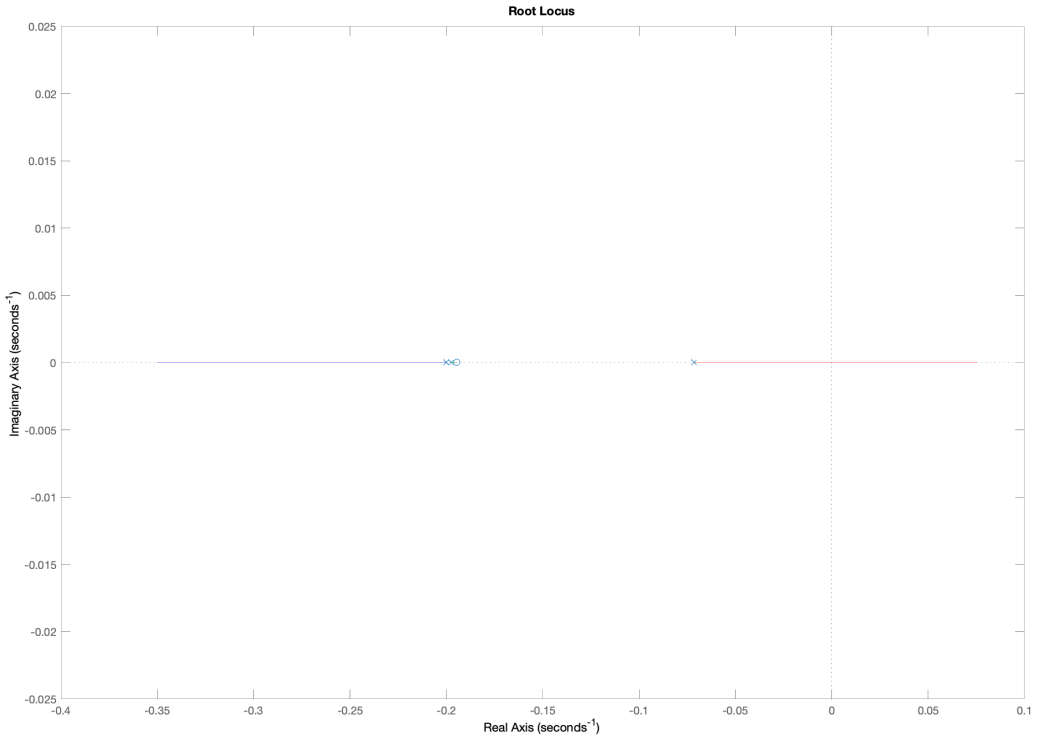
Comparing this conclusion with the assumptions made for the linearization allows us to identify the cause for this instability. The number of exposed individuals is also related to the number of suceptible individuals and the linearized model **assumes the number of suceptible individuals will remain approximately around the operating point** S_e. In the non-linear model the number of people in *S* decreased when an individual moved to the E, I or J groups, but the linearization uses the constant S_e to calculate its rate of change at any point in time, even if S_δ indicates that there are no remaining individuals in this caregory. This assumption effectively injects an infinite amount of people into the model by producing negative values for the suceptible population, which in turn causes the J curve to behave in an unstable manner if the amount of injected people is greater than the amount leaving the model through either recovery or death.

However, this results also indicates that for a small enough value of S_e the system should be stable, as the number of equilibirum suceptible individuals would not be enough to replace the people exiting the model (either through recovery or death), which would exactly reproduce the conditions necessary for the model stability in the non-linear case (even though the constant value of S_e could still artificially increase the total proportion of the population by generating negative population values).

For this reason we perform a Root locus with respect to S_e

$$H(s) = \frac{294000s^2 + (79800 - 58800\,u_e)s + (42000 - 42000\,u_e + 11760\,S_e\,u_e - 11760\,S_e)}{1 + S_e \dfrac{(12045\,u_e - 12054)s + (2345\,u_e - 2345)}{294000s^3 + (137900 - 58800\,u_e)s^2 + (19970 - 15820\,u_e)s + (830 - 830\,u_e)}}$$

```
% Define arbitrary equilibrium u to obtain root locus
u_e = 0;
num = [(-12054+12054*u_e) (-2345+2345*u_e)];
denom = [294000 (137900-58800*u_e) (19970-15820*u_e) (830-830*u_e)];

% Define function and plot
L = tf(num,denom);
rlocus(L,k)
```



This Root locus effectively shows the desired behavior as choosing an equilibrium population below ≈ 0.345 leads to a stable system (it is interesting to note that this value also corresponds to $q+\rho_2 = \frac{83}{420} \approx 0.345$)

Update after Professor's announcement

Professor Tabuada suggested we choose an equilibrium pair where we would like to opearte **at the end** of the pandemic in order to obtain a better assessment of pole dependence on u_e. At the end of the pandemic the suceptible population should logically be close to 0 and this suggestion supports the conclusions drawn above, as only low values of S_e will result in a stable system. In this scenario the most natural operating point to evaluate the system would be $S_e = 0$

$$H(s) = \frac{5880s^2 + (1596 - 1176\,u_e)s + 84(1-u_e)}{(14s+1)(420s+83)\left(s+\frac{1}{5}-\frac{u_e}{5}\right)}$$

However, the problem with this approach lies in the fact that setting $S_e = 0$ causes the following B matrix:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{u_e}{5}-\frac{1}{5} & 0 & 0 & 0 & 0 \\ 0 & \frac{4}{25}-\frac{4\,u_e}{25} & -\frac{1}{14} & 0 & 0 & 0 \\ 0 & \frac{1}{25}-\frac{u_e}{25} & 0 & -\frac{83}{420} & 0 & 0 \\ 0 & 0 & \frac{1}{14} & \frac{1}{21} & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{20} & 0 & 0 \end{pmatrix}$$

The matrix suggests the value of S_δ will always remain constant as its derivative is always zero, and this is not a useful representation of our model. Therefore, we will use a low value of $S_e = 0.1$ for the next problem, in order to maitain stability, while using the intuition derived from the first root locus to modify u_e.

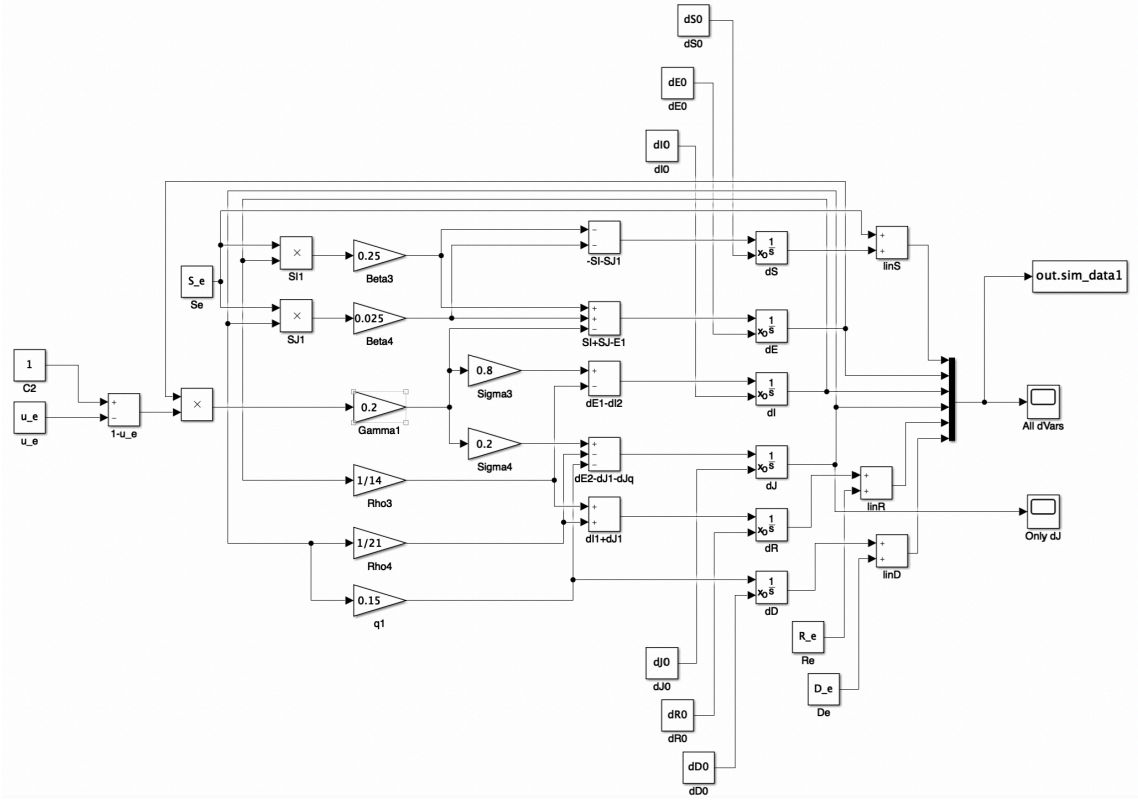
Problem 5

Simulate the linearized model for different values of u and for the initial condition:

$S(0) = 0.8,\ E(0) = 0.145,\ I(0) = 0.05,\ J(0) = 0.005,\ R(0) = 0,\ D(0) = 0$

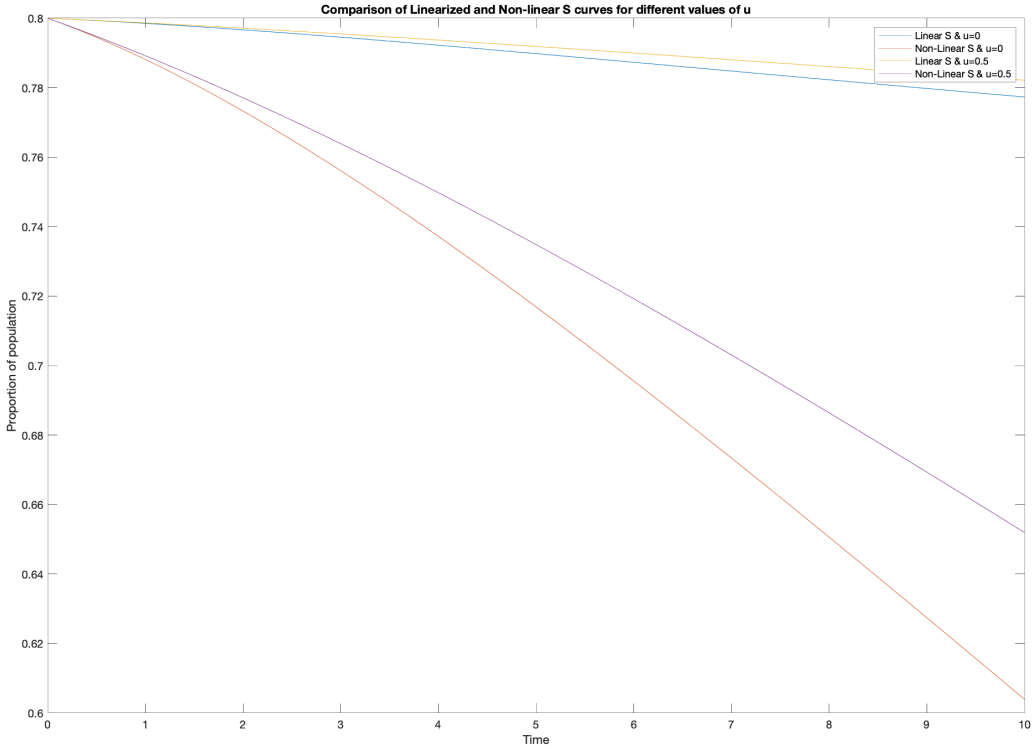
Comment on what you observe and how it relates to the poles of the transfer function in the preceding question. Compare the simulation results of the linearized model with those of the nonlinear model. Explain the differences.

```
imshow(imread("Linearized_model.bmp"))
```

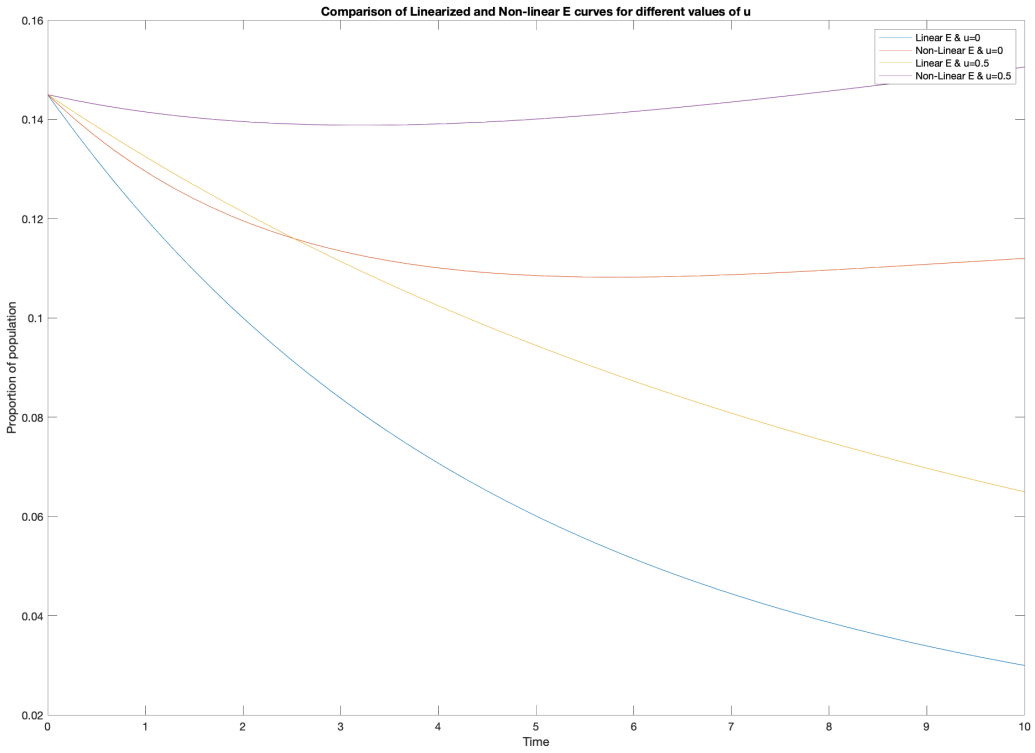



```
%Set up initial constants for non-linear model
sim_t = 10.0;
S0 = 0.8;
I0 = 0.05;
E0 = 0.145;
J0 = 0.005;
R0 = 0;
D0 = 0;
u_val = 0;

% Set up initial constants for linear model
u_e = u_val;
S_e = 0.1;
D_e = 0.1;
R_e = 0.8;
dS0 = S0-S_e;
dI0 = I0-0;
dE0 = E0-0;
dJ0 = J0-0;
dR0 = R0-R_e;
dD0 = D0-D_e;


% The combined model is just a single Simulink file
% containing both the linearized and non-linear models
% to simplify model simulation and comparison
```

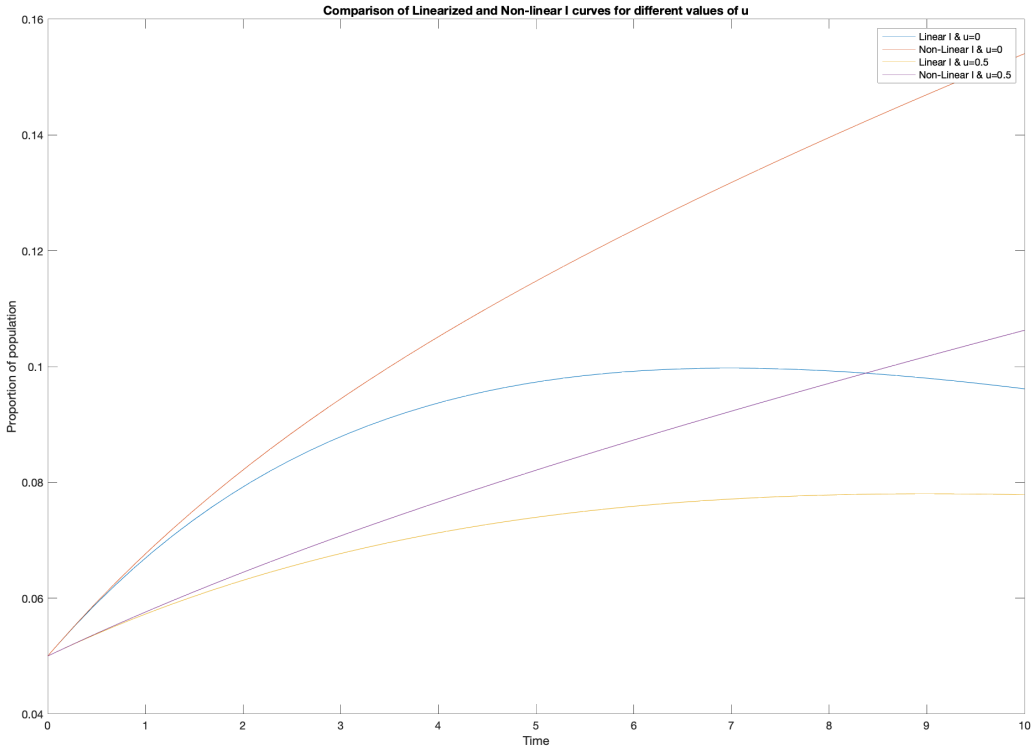
```
% Simulate model with given ICs
simOut = sim('Combined_model',sim_t);
sim_data_lin1 = simOut.get("sim_data1");
sim_data_nlin1 = simOut.get("sim_data");

u_val = 0.5;
u_e = u_val;

% Simulate model with given ICs
simOut = sim('Combined_model',sim_t);
sim_data_lin2 = simOut.get("sim_data1");
sim_data_nlin2 = simOut.get("sim_data");

t = sim_data_lin1.Time;
data_lin1 = sim_data_lin1.Data;
data_nlin1 = sim_data_nlin1.Data;
data_lin2 = sim_data_lin2.Data;
data_nlin2 = sim_data_nlin2.Data;


plot(t, [data_lin1(:,1) data_nlin1(:,1) data_lin2(:,1) data_nlin2(:,1)])
title("Comparison of Linearized and Non-linear S curves " + ...
    "for different values of u")
xlabel("Time")
ylabel("Proportion of population")
legend("Linear S & u=0", "Non-Linear S & u=0", ...
    "Linear S & u=0.5", "Non-Linear S & u=0.5" )
```



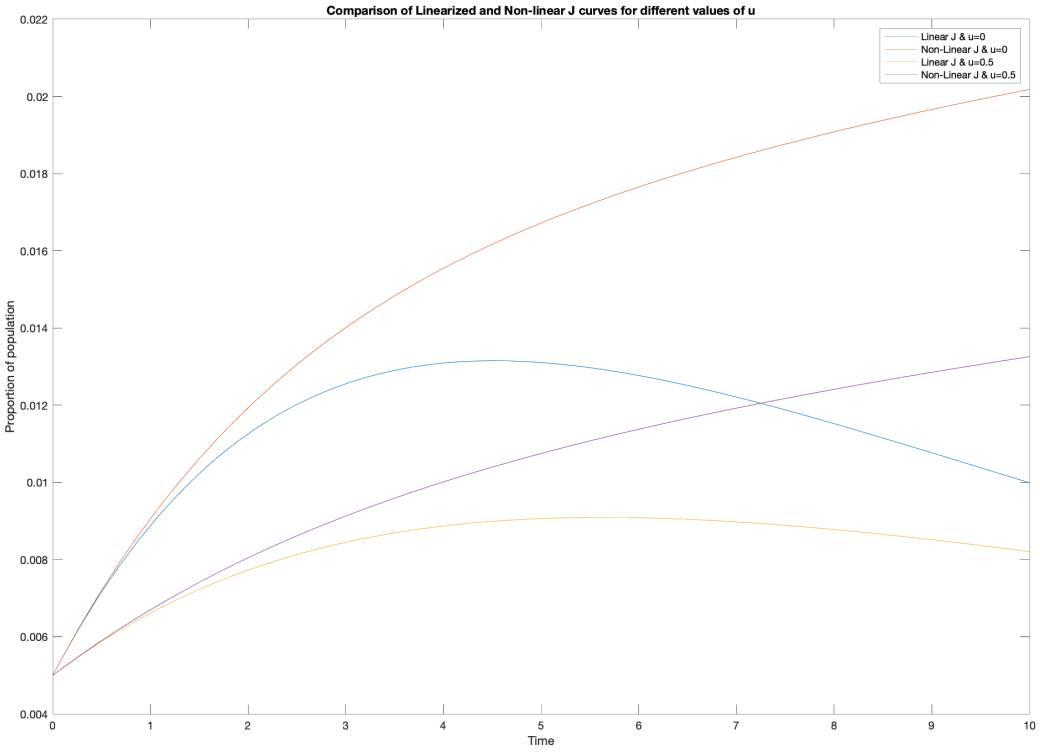
```
plot(t, [data_lin1(:,2) data_nlin1(:,2) data_lin2(:,2) data_nlin2(:,2)])
title("Comparison of Linearized and Non-linear E curves " + ...
    "for different values of u")
xlabel("Time")
ylabel("Proportion of population")
legend("Linear E & u=0", "Non-Linear E & u=0", ...
    "Linear E & u=0.5", "Non-Linear E & u=0.5" )
```



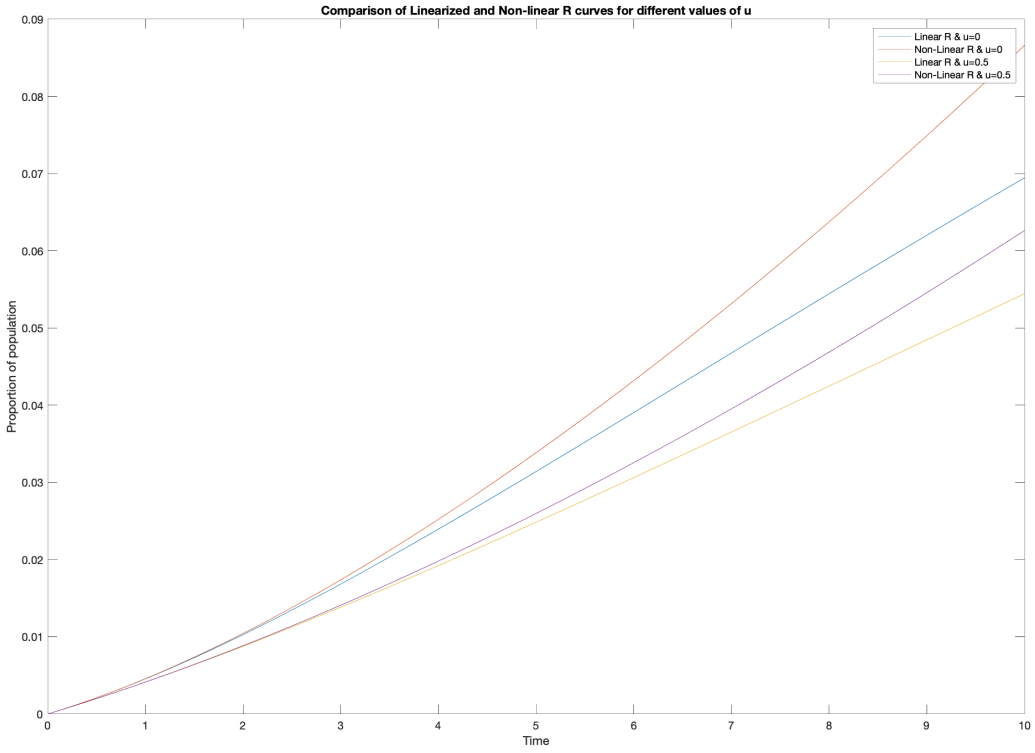

```
plot(t, [data_lin1(:,3) data_nlin1(:,3) data_lin2(:,3) data_nlin2(:,3)])
title("Comparison of Linearized and Non-linear I curves " + ...
    "for different values of u")
xlabel("Time")
ylabel("Proportion of population")
legend("Linear I & u=0", "Non-Linear I & u=0", ...
    "Linear I & u=0.5", "Non-Linear I & u=0.5" )
```



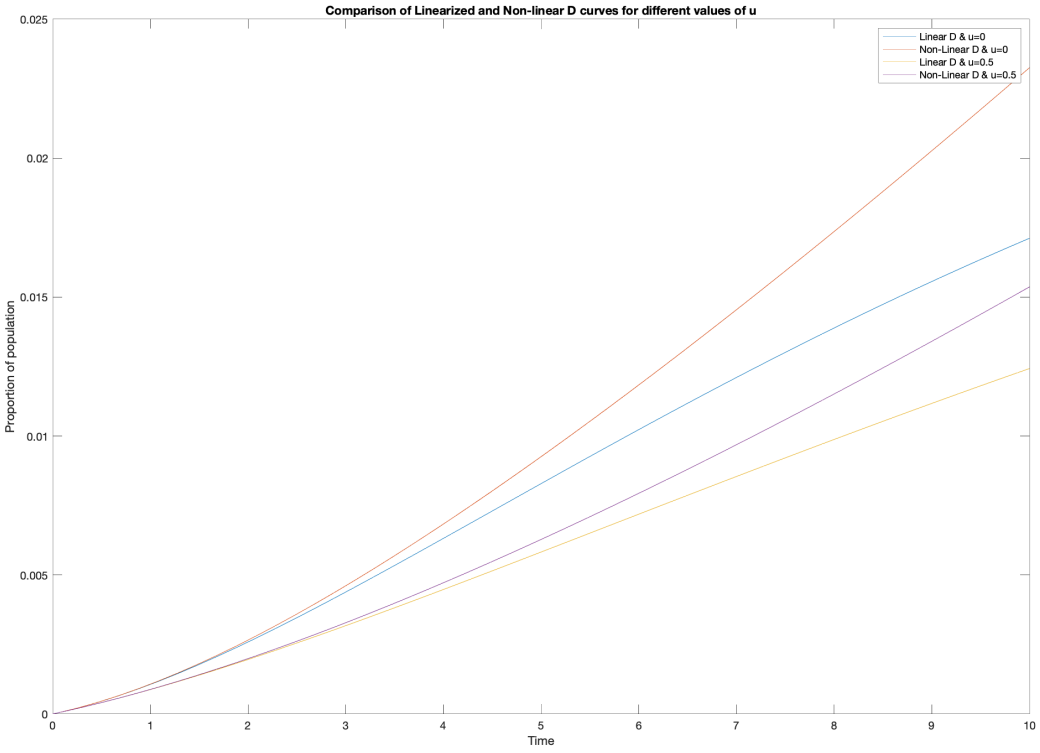

```
plot(t, [data_lin1(:,4) data_nlin1(:,4) data_lin2(:,4) data_nlin2(:,4)])
title("Comparison of Linearized and Non-linear J curves " + ...
    "for different values of u")
xlabel("Time")
ylabel("Proportion of population")
legend("Linear J & u=0", "Non-Linear J & u=0", ...
    "Linear J & u=0.5", "Non-Linear J & u=0.5" )
```




```
plot(t, [data_lin1(:,5) data_nlin1(:,5) data_lin2(:,5) data_nlin2(:,5)])
title("Comparison of Linearized and Non-linear R curves " + ...
    "for different values of u")
xlabel("Time")
ylabel("Proportion of population")
legend("Linear R & u=0", "Non-Linear R & u=0", ...
    "Linear R & u=0.5", "Non-Linear R & u=0.5" )
```




```
plot(t, [data_lin1(:,6) data_nlin1(:,6) data_lin2(:,6) data_nlin2(:,6)])
title("Comparison of Linearized and Non-linear D curves " + ...
    "for different values of u")
xlabel("Time")
ylabel("Proportion of population")
legend("Linear D & u=0", "Non-Linear D & u=0", ...
    "Linear D & u=0.5", "Non-Linear D & u=0.5" )
```



All resulting curves seem to share the same general features for small values of *t*. For the reasons discussed in the previous problem the curves start diverge significantly after a sufficiently large amount of time. Furthermore, an increase in the value of *u* has the expected effect on the model as the growth of the *J* curve slows down significantly in both cases, even if there is a notorius offset between the two models.

These results suggests some interesting conclusions regarding controller design, as changing *u* seems to have a drastic impact on the behavior of the *J* curve even though the linearization of the model showed the effect of the input *u* was completely zeroed out. This clearly supports the use of a switched linear system for control purposes, as the operating point must be changed throughout execution to effectively use the intuition derived from the linearized approximation.

Problem 6

Design a controller for the non-linear model that satisfies the following specifications:

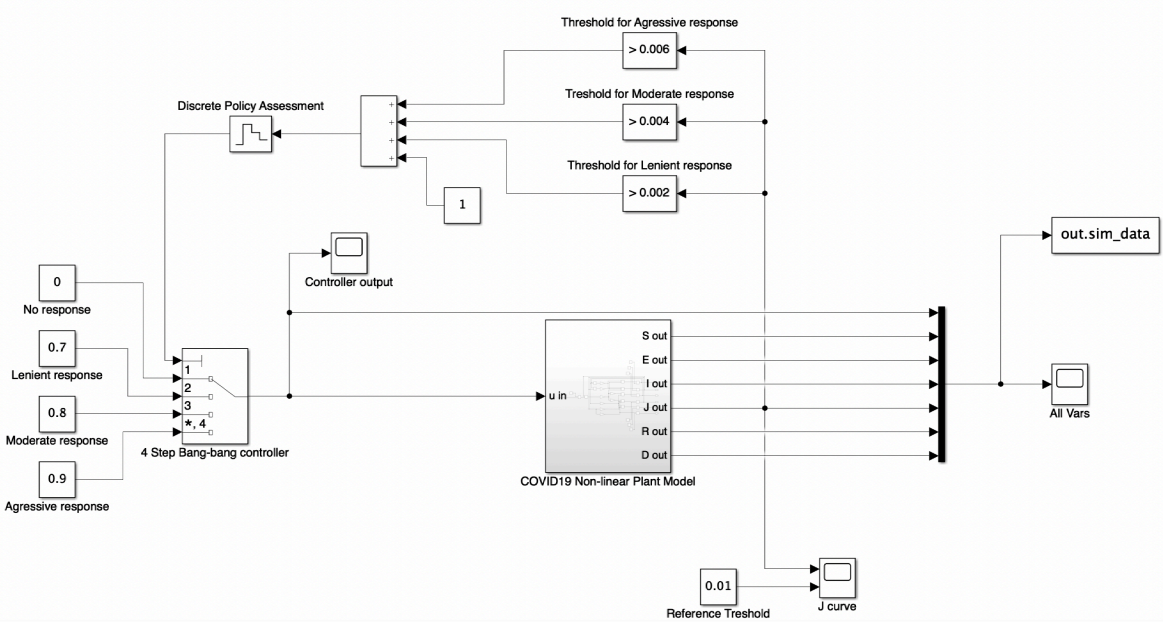
1. $0 \leq u(t) \leq 0.9, \forall t$
2. $J(t) < 0.1, \forall t$
3. $u(t)$ is applied at discrete time intervals simulating policy changes
4. Make $u(t)$ as small as possible
5. Switch $u(t)$ as few times as possible

My implementation uses a 4 port switch combined with comparison blocks to implement a 4 step bang-bang controller where the input to the system (*u*) changes discontinuously between 4 predefined values. In order to

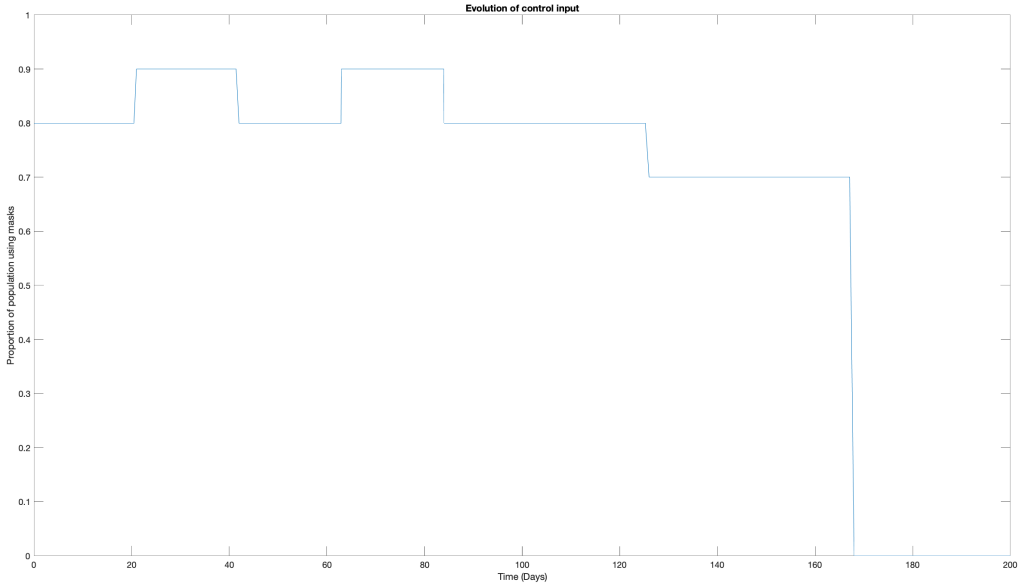
enforce the discrete requirement I combined the comparison blocks with a zero-order hold, this implementation would mimic the government assessing the situation every 21 days (3 weeks) and deciding whether a policy change is required.

In my controller Simulink file the whole non-linear model has been collapsed into a subsystem block.

```
imshow(imread("Controller.bmp"))
```



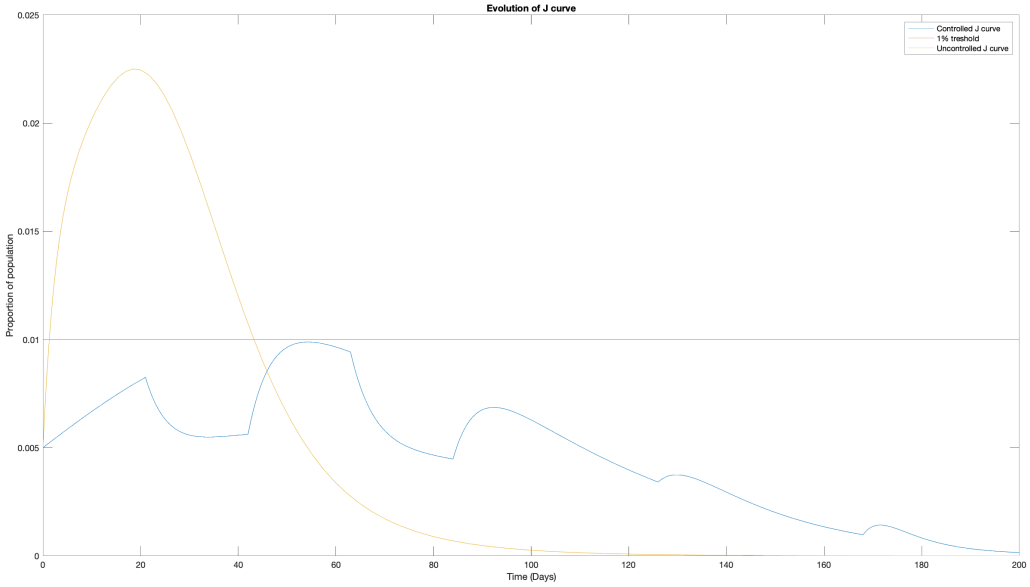

```
%Set up initial constants for non-linear model
sim_t = 200.0;
S0 = 0.8;
I0 = 0.05;
E0 = 0.145;
J0 = 0.005;
R0 = 0;
D0 = 0;

% Simulate model with given ICs
simOut = sim('Controller',sim_t);
control_data = simOut.get("sim_data");

t = control_data.Time;
data = control_data.Data;

% Plot control input evolution
plot(t, data(:,1))
title("Evolution of control input")
xlabel("Time (Days)")
ylabel("Proportion of population using masks")
ylim([0,1])
```



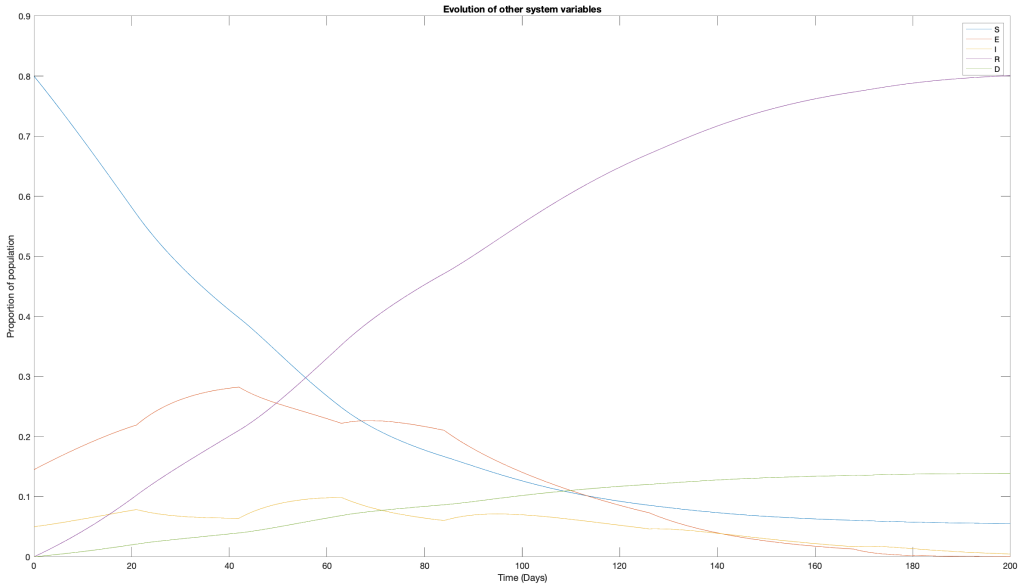

```
% Compute average proportion of population using masks
av = sum(data(:,1))/length(t)
```

```
av = 0.6832
```

```
% Compute 1% threshold
thresh = repmat( 0.01, [length(t),1] );
% Plot J curve with threshold
plot(t, [data(:,5) thresh])
hold on
plot(t1, J1)
hold off
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("Controlled J curve", "1% treshold", "Uncontrolled J curve")
title("Evolution of J curve")
```




```
% Plot S curve
plot(t, [data(:,2) data(:,3) data(:,4) data(:,6) data(:,7)])
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("S", "E","I","R","D")
title("Evolution of other system variables")
```




```
fprintf("Steady state S = %f",data(end,2))
```

```
Steady state S = 0.055323
```

```
fprintf("Steady state R = %f",data(end,6))
```

```
Steady state R = 0.800680
```

```
fprintf("Steady state D = %f",data(end,7))
```

```
Steady state D = 0.138670
```

Comparing the results from the controlled model and the first simulated case in Problem 1, we can see that the steady state value of the S,R, and D variables is almost identical. This suggests that our control techniques do not affect the final state achieved by the system, but rather the shape of the curves.

However, in Problem 2 we established that a great number of deaths in a real-life scenario would be caused by the saturation of the healthcare system's capacity to treat seriously ill patients. In this case, the control requirements simulate a country with just enough ICUs to treat 1% of its population simultaneously. The peak of the J curve for the uncontrolled model shows about 2.25% of the population being seriously ill, which implies that more than half of the patients would be unable to get proper treatment, causing a sharp increase in mortality and generating a significant rise in overall death toll.

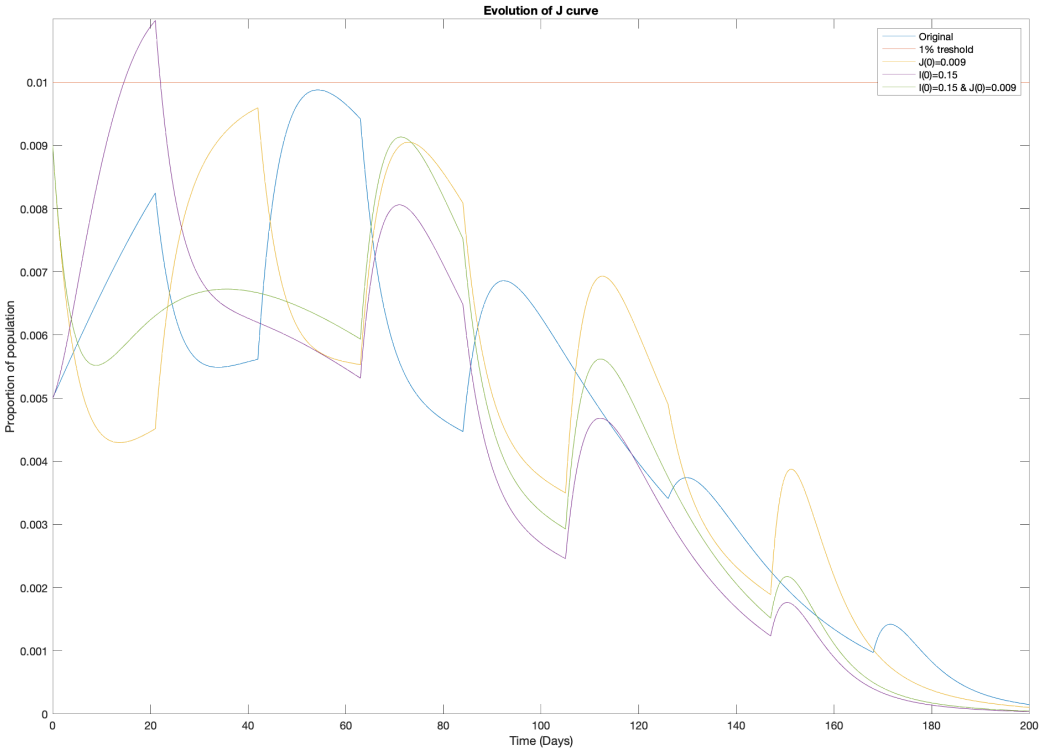
Moreover, the control system presented requires an average of 68% of people wearing masks throughout simulation period, which seems reasonable considering that an agressive response (90% of people using masks) was only enforced for 6 weeks (2 sets of 3 weeks/21 days). The controller also produced relatively few ploicy changes, as the percentage of population using masks only changed 6 times in the 200 day simulation. All in all, the model produced results that would be applicable in a real-life scenario.

Problem 7

How does your controller perform when you change the initial conditions? How would you change it so that it works for any initial condition?

```
%Set up initial constants for non-linear model
sim_t = 200.0;
S0 = 0.8;
I0 = 0.05;
E0 = 0.145;
J0 = 0.005;
R0 = 0;
D0 = 0;

% Simulate model with given ICs
simOut = sim('Controller',sim_t);
control_data = simOut.get("sim_data");

t1 = control_data.Time;
data1 = control_data.Data;

S0 = 0.796;
J0 = 0.009;

% Simulate model with given ICs
simOut = sim('Controller',sim_t);
control_data = simOut.get("sim_data");
```

```
t2 = control_data.Time;
data2 = control_data.Data;

S0 = 0.7;
J0 = 0.005;
I0 = 0.15;

% Simulate model with given ICs
simOut = sim('Controller',sim_t);
control_data = simOut.get("sim_data");

t3 = control_data.Time;
data3 = control_data.Data;

S0 = 0.696;
J0 = 0.009;
I0 = 0.15;

% Simulate model with given ICs
simOut = sim('Controller',sim_t);
control_data = simOut.get("sim_data");

t4 = control_data.Time;
data4 = control_data.Data;

% Compute 1% threshold
thresh = repmat( 0.01, [length(t),1] );
% Plot J curve with threshold
plot(t1, [data1(:,5) thresh])
hold on
plot(t2, data2(:,5))
plot(t3, data3(:,5))
plot(t4, data4(:,5))
hold off
xlabel("Time (Days)")
ylabel("Proportion of population")
legend("Original", "1% treshold", "J(0)=0.009", ...
    "I(0)=0.15","I(0)=0.15 & J(0)=0.009")
ylim([0,0.011])
title("Evolution of J curve")
```



Experimenting with different initial conditions shows that the controller designed is robust to changes in the initial number of seriously ill patients, as long as this initial condition does not start by violating the 1% threshold. The model also seem to be uncapable of controlling the J curve when the number of infected individuals rises sharply (10% more initially infected people), causing a relatively brief but unsatisfactory peak above 1%.

However, combining both changes in initial conditions $J(0) = 0.009$ and $I(0) = 0.15$ produces a *J* curve that satisifes the requirments. This results suggests that the undesirable behavior identified in the previous case was not generated by an inability of the controller to handle an increase in infected individuals, but rather because of the discretization applied to simulate policy changes. In this scenario, the controller applied an initial policy that was too lenient to properly handle the situation, and wasn't able to correct its approach until 3 weeks later when the peak had already occurred. There are 3 possible solutions that could be applied to increase the robustness of the system an potentially meet most requirements for any given initial condition:

1. Shorten the time delay between policy changes: This is the simplest solution and it would allow the model to react more quickly when presented with rapidly evolving situations. Nonetheless, it conflicts with the optimization objective of changing the policy relatively infrequently, as people would be unable/ unwilling to comply with quickly shifting policies.
2. Lower thresholds for triggering each response/Increase magnitude of responses: This solution is also relatively simple even though it does require some redesigining, and in the limiting case ($u(t) = 0.9 \text{ for } J(t) > 0$) it has the potential of dealing with most initial conditions. However, the solution conflicts with the optimization objective of making $u(t)$ as small as possible, given that the population would be subjected to unnecessarily drastic responses in many cases.

3. Consider other variables/rates of change: This solution has the greatest potential to satisfy the constraints and optimization challenges. As seen above, the behavior of the J curve is not solely determined by its present state but also by the state of the other variables. The magnitude of the response could be fine tuned by using these considerations as they provide additional information about how the system will evolve in the near future. This approach could be further improved by considering the rate of change of these variables and analyzing how they affect the J curve and, consequently, the required response for any given situation.